Formularios y Modals

En este ejercicio vamos a introducir nuestro primer formulario en la aplicación de Gaztaroa. Además, crearemos un *Modal* y aprenderemos a hacerlo aparecer y desaparecer atendiendo a la interacción con el usuario.

1. Formularios

Nuestro formulario hará uso del Picker para seleccionar la edad y de Date Picker, para seleccionar el día y la hora en las que quiere realizar la prueba de esfuerzo. Para ello, se debe instalar:

yarn add @react-native-picker/picker
yarn add @react-native-community/datetimepicker

Se crea un nuevo archivo en la carpeta componentes llamado PruebaEsfuerzoComponent.js.

Para seleccionar la edad en nuestro formulario se hace uso de la función Picker. Para indicar si una persona está federada o no se utiliza un switch, lo que viene a ser un checkbox. Para poder abrir el cuadro de diálogo para seleccionar la fecha y la hora se utiliza el DatePicker. Finalmente, un Button para enviar la información del formulario.

Por ahora todos los datos se almacenan en el estado del componente y no en Redux.

La función gestionarReserva(), se encarga de mandar el estado y lo saca por consola y resetea el estado.

Hay que tener en cuenta que para visualizar nuestro componente en la app se debe actualizar el navegador.

2. Modals

El modal se incorpora en este componente. Se deberá realizar alguna actualización de lo anterior.

Para comenzar se añade al estado del componente la variable boolena showModal, esto indica si el modal se está mostrando o no.

Se crea ahora una nueva función resetForm(), para resetear el estado de todas las variables. Por tanto ahora la función gestionarReserva(), ya no hará falta que lo resetee.

La función toogleModal(), invierte la variable de estado *showModal* en función del estado del *Modal.* Esta función será llamada en la función gestionarReserva().

A continuación, se añade el modal, es una forma básica de presentar contenido sobre una vista adjunta.

```
 <Modal animationType = {"slide"} transparent = {false}
                visible = {this.state.showModal}
                onDismiss = {() => {this.toggleModal(); this.resetForm();
}}
                onRequestClose = {() => {this.toggleModal(); this.resetFo
rm();}}>
                <View style = {styles.modal}>
```

```
                    <Text style = {styles.modalTitle}>Detalle de la reser
va</Text>
                    <Text style = {styles.modalText}>Edad: {this.state.ed
ad}</Text>
                    <Text style = {styles.modalText}>Federado?: {this.sta
te.federado ? 'Si' : 'No'}</Text>
                    <Text style = {styles.modalText}>Día y hora: {new Int
l.DateTimeFormat('default', { weekday: 'long', year: 'numeric', month: 'n
umeric', day: 'numeric', hour: 'numeric', minute: 'numeric'}).format(Date
.parse(this.state.fecha))}</Text>
                    <Button
                        onPress = {() =>{this.toggleModal(); this.resetFo
rm();}}
                        color={colorGaztaroaOscuro}
                        title="Cerrar"
                        />
                </View>
            </Modal>
```

El prop visible que indica si el modal es visible o no. El prop onDismiss permite pasar una función que se llamará una vez que se haya descartado el modal, en este caso dos la que resetea los estados y la función invierte el estado showmodal, esto en iOS para Android estaría el prop onRequestClose. También estas funciones serán llamadas al presionar el botón Cerrar.

Finalmente quedaría así: